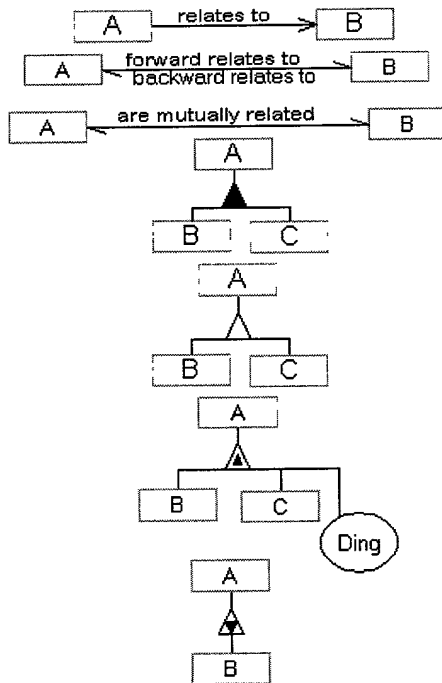


## Appendix A



**A relates to B.**

*Forward structure*

**A forward relates to B and B backward relates to A.**

*Bi-directional structure sentence*

**A and B are mutually related.**

*Homologous*

**A consists of B and C.**

*Aggregation*

**B and C are As.**

*Specialization sentence*

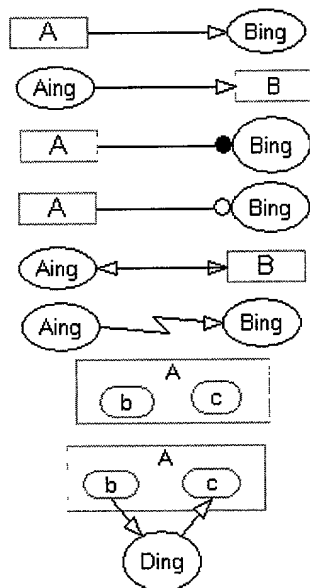
**A exhibits B and C, as well as Ding.**

*Exhibition*

**B is an instance of A.**

*Instantiation*

### Behavior and state sentences



**Bing consumes A.**

*Consumption sentence*

**Aing yields B.**

*Result sentence*

**A handles Bing.**

*Agent sentence*

**Bing requires A.**

*Instrument sentence*

**Aing affects B.**

*Effect sentence*

**Aing invokes Bing.**

*Invocation sentence*

**A can be b or c.**

*State enumeration sentence*

**Ding changes A from b to c.**

*Change sentence*

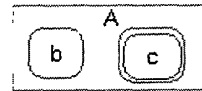
**Fixing changes Car from broken to fixed.**

*Change sentence*



**A is initially b.**

*Initial state specification*



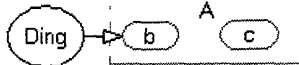
**A is ultimately c.**

*Ultimate state specification sentence*



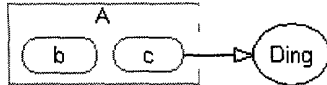
**A is initially b and ultimately c.**

*Initial and ultimate state specification*



**Ding yields b A.**

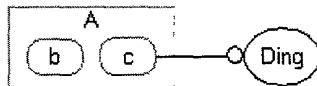
*State-specified result sentence*



**Ding consumes c A.**

*sentence*

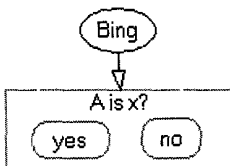
*State-specified consumption*



**Ding requires c A.**

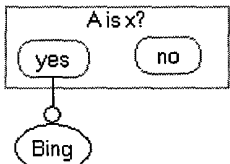
*State-specified instrument sentence*

### Boolean sentences



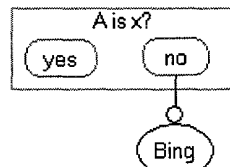
**Bing determines whether A is x.**

*Determination sentence*



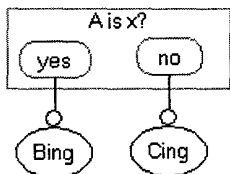
**Bing occurs if A is x.**

*Condition sentence*



**Bing occurs if A is not x.**

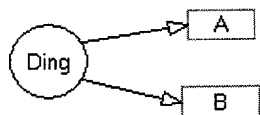
*Negative condition sentence*



**Bing occurs if A is x, otherwise Cing occurs.**

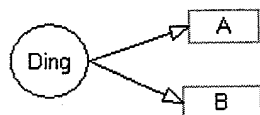
*Compound condition*

## Logical operation and probabilistic sentences

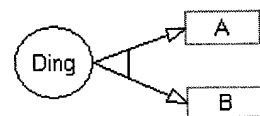


**Ding yields A and B.**

*And result sentence*

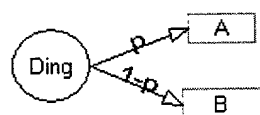


**Ding yields either A or B.** *Exclusive or result sentence*



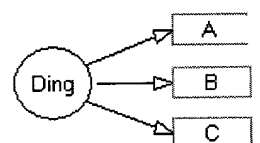
**Ding yields A or B.**

*Or result sentence*

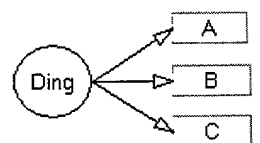


**Ding yields either A, with probability  $p$ , or B, with probability  $1-p$ .**

*Probabilistic exclusive or result sentence*

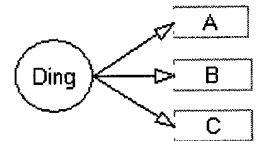


**Ding yields A, B, and C.** *And result sentence*



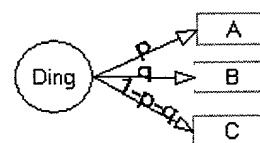
**Ding yields either A or B or C.**

*Exclusive or result sentence*



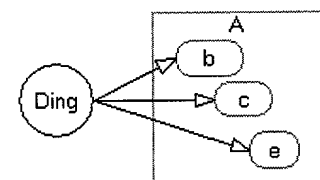
**Ding yields A or B or C.**

*Or result sentence*



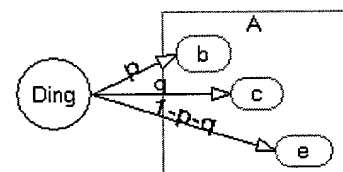
**Ding yields either A, with probability  $p$ , or B, with probability  $1-p$ .**

*Probabilistic result sentence*



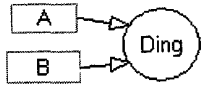
**Ding yields either b A or c A or e A.**

*Exclusive or result sentence*

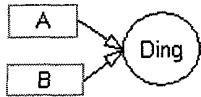


**Ding yields either b A, with probability  $p$ , or c A, with probability  $q$ , or e A, with probability  $1-p-q$ .**

*Probabilistic result sentence*



**Ding consumes A and B.** *And consumption sentence*



**Ding consumes either A or B.** *Exclusive or consumption sentence*

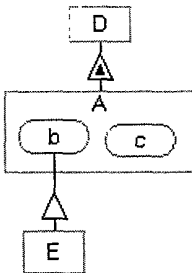


**Ding consumes A or B.** *Or consumption sentence*



**Ding consumes either A, with probability  $p$ , or B, with probability  $1-p$ .**  
*Probabilistic consumption sentence*

#### Qualification sentences



**E is a D, the A of which is b.** *Compound condition sentence*

**Airplane is a vehicle, the Traveling Medium of which is water.**

## Appendix B

### Notational Conventions

- (1) `/character` denotes an ASCII character, e.g., `/period is "."` `/space is " "`, `/comma is ","`.
- (2) `||` denotes string concatenation.
- (3) `(pattern)?` means optional (zero or one) repetitions of pattern.  
`(pattern)*` means zero or more repetitions of pattern.  
`(pattern)+` means at least one repetition of pattern.  
`(pattern)n+` means at least n repetitions of pattern.

(4) `(?!pattern)` is a Perl regular expression convention called a zero-width negative lookahead assertion. For example `/foo(?!bar)/` matches any occurrence of `foo` that isn't followed by `bar`. So this means that *Object-name* must not end with the `ing` suffix, or, if it does end with `ing` it must be followed by the reserved word `Object`. Example: `Ready Casting Object`.

Likewise, a process name must end with `ing`, and if it does not, it must be followed by the reserved word `Process`. Example: `Initiation Process`.

- (5) `a → a | A` underlined non-capital letter designates a non-capital or capital letter.

(6) A production right hand side which contains the phrase *English syntax for* means that the grammar must follow the syntax of English. Example: *English syntax for Irregular-Plural-Form*

- (7) `A | B → C` is shorthand for `A → C and B → C`

(8) `↓` is string stripping - the reverse operation for string concatenation. `A↓a` means the string `A` from which the string `a` was stripped. Example:

*Boolean-object-name* `→ Boolean-assertion|| /question-mark`

*Boolean-assertion* `→ Boolean-object-name↓/question-mark`

- (9) Non bold Arial words denote reserved terminal. **Bold Arial** words denote non-reserved terminals.

- (10) `/* This is a comment */`

## Production rules

**S** → *Formal-sentence* /period

*Formal-sentence* → *Generic-attribute-sentence*

| *Structure-sentence* | *State-sentence*  
| *Behavior-sentence* | *Control-sentence*  
| *Complexity-management-sentence*

*Generic-attribute-sentence* → *Thing-name* is (a | an) *Generic-attribute-list* (**Object** | **Process**)

*Generic-attribute-list* → *Generic-attribute* (and *Generic-attribute*)

*Generic-attribute* → Physical | Environmental | Transient | Natural |  $\phi$

*/\* Physical is the non-default value of Essence,  
the default value of which is Informatical. \*/*  
*/\* Environmental is the non-default value of Affiliation,  
the default value of which is Systemic. \*/*  
*/\* Transient is the non-default value of Persistence,  
the default value of which is Persistent. \*/*  
*/\* Natural is the non-default value of Origin,  
the default value of which is Artificial. \*/*

*Structure-sentence* → *Tagged-strucutre-sentence* | *Fundamental-strucutre-sentence*

*Tagged-strucutre-sentence* → *Forward-strucutre-sentence*

| *Bi-directional-strucutre-sentence*  
| *Homologous-strucutre-sentence*

*Forward-strucutre-sentence* → *Source-object-name* *Forward-tag* *Destination-object-name*

*Backward-strucutre-sentence* → *Destination-object-name* *Backward-tag* *Source-object-name*

*Bi-directional-strucutre-sentence* → *Forward-strucutre-sentence* and *Backward-strucutre-sentence*

*Homologous-strucutre-sentence* → *Object-name* and *Object-name* are *Homologous-tag*

*Source-object-name* | *Destination-object-name* → *Object-name*

*Forward-tag* | *Forward-tag* *Backward-tag* | *Homologous-tag* → *Non-capitalized-phrase*

*Capitalized-phrase* → *Capitalized-word* (/space)\* (*Capitalized-word* (/space))\*

*Non-capitalized-phrase* → *Non-capitalized-word* (/space)\* (*Non-capitalized-word* (/space))\*

*Capitalized-word* → *Capital-letter* (*Character*)\*

*Non-capitalized-word* → *Letter* (*Character*)\*

*Capital-letter* → [A..Z]

*Character* → *Letter* | *Digit* | - | &

*Letter* → [a..z, A..Z]

*Digit* → [0..9]

*Single* → a | an | an optional | at least one

*Plural* → optional | many | *Number*

*Object-name* → *Single*? *Capitalized-phrase* (?!ing)  
| *Single*? *Capitalized-phrase*||ing *Object*  
| *Plural*? *Object-plural-name*

*Object-plural-name* → *Capitalized-phrase*(?!ing)||s  
| *English syntax for Irregular-Plural-Form*  
| *Capitalized-phrase*(?!ing)||ing *Objects*

*Object-list* → (*Object-name*/comma/space)<sup>2+</sup> and *Object-name* |  
*Object-name* and *Object-name* | *Object-name*

*Process-name* → *Single*? *Capitalized-phrase*||ing)  
| *Single*? *Capitalized-phrase*(?!ing) *Process*  
| *Plural*? *Process-plural-name*

*Process-plural-name* → *Capitalized-phrase*(?!ing)||ings  
| *Capitalized-phrase*(?!ing) *Processes*

*Process-list* → (*Process-name* /comma /space)<sup>2+</sup> and *Process-name* |  
*Process-name* and *Process-name* | *Process-name*

*Thing-list* → *Object-list* | *Process-list* | *Process-and-Object-list* | *Object-and-Process-list*

*Process-and-Object-list* → *Process-list*/comma as well as *Object-list*

*Object-and-Process-list* → *Object-list*/comma as well as *Process-list*

*Number* → *Integer* | *Real*

*Integer* → [(two | 2) .. (ten, 10), 11 .. 999999999999]

*Real* → (*Digit*)\* /period (*Digit*)\*

*Fundamental-structure-sentence* → *Aggregation-sentence*  
| *Exhibition-sentence*  
| *Specialization-sentence*

| *Instantiation-sentence*

*Aggregation-sentence* → *Object-name* consists of *Object-list*

| *Process-name* consists of *Process-list*

*Exhibition-sentence* → *Thing-name* exhibits *Object-list*

| *Object-list* /*comma* and *Process-list*

| *Process-list*

*Specialization-sentence* → *Object-specialization-sentence* | *Process-specialization-sentence*

*Object-specialization-sentence* → *Object-name* is (a | an) *Object-name*

| *Object-list* are *Object-name*

*Process-specialization-sentence* → *Process-name* is (a | an) *Process-name*

| *Process-list* are *Process-name*

*Instantiation-sentence* → *Object-Instantiation-sentence* | *Process-Instantiation-sentence*

*Object-Instantiation-sentence* → *Object-name* is an instance of *Object-name*

| *Object-list* are instances of *Object-name*

*Process-Instantiation-sentence* → *Process-name* is an instance of *Process-name*

| *Process-list* are instances of *Process-name*

*State-sentence* → *State-enumeration-sentence*

*State-enumeration-sentence* → *Object* can be *State-list*

*State-list* → (*State-name*/*comma*/*space*)<sup>2+</sup> or *State-name* |

*State-name* or *State-name* | *State-name*

*Behavior-sentence* → *Enabling-sentence* | *Transformation-sentence* | *Invocation-sentence*

*Enabling-sentence* → *Agent-sentence* | *Instrument-sentence*

*Agent-sentence* → *Object-name* handles *Process-list* | *Object-list* handle *Process-list*

*Instrument-sentence* → *Process-name* requires *Object-list* | *Process-list* require *Object-list*

*Transformation-sentence* → *Consumption-sentence* | *Result-sentence*

| *Effect-sentence* | *Change-sentence*

*Consumption-sentence* → *Process-name* consumes *Object-list*

*Result-sentence* → *Process-name* yields *Object-list*

*Effect-sentence* → *Process-name* affects *Object-list* | *Process-list* affect *Object-list*

*Change-sentence* → *State-change-sentence* | *Value-change-sentence*

*State-change-sentence* →



*Process-name changes Object-name from Source-state-name to Destination-state-name*

| *Source-state-change-sentence*

| *Destination-state-change-sentence*

*Source-state-change-sentence* → *Process-name changes Object-name from Source-state-name*

*Destination-state-change-sentence* →

*Process-name changes Object-name to Destination-state-name*

*Source-state-name* | *Destination-state-name* → *State-name*

*State-name* → *Non-capitalized-word*

*Value-change-sentence* → *Process-name changes Object-name from Value-name to Value-name*

*Value-name* → *Number* | *Capitalized-word*

*Invocation-sentence* → *Process-name invokes Process-name*

*Control-sentence* → *Determination sentence*

| *Condition-sentence* | *Negative-condition-sentence*

| *Compound-condition-sentence* | *Case-sentence*

*Determination sentence* → *Process-name determines whether Boolean-assertion*

*/\* Boolean-object-name → Boolean-assertion|| ?*

*Boolean-assertion → Boolean-object-name↓? /*

*Boolean-assertion* is *Boolean-object-name* from which the question-mark which is mandatory at the end of *Boolean-object-name* was stripped. *\*/*

*Boolean-assertion* → *Opening-Boolean-phrase* is *Closing-Boolean-phrase*

*Negated-Boolean-assertion* → *Opening-Boolean-phrase* is not *Closing-Boolean-phrase*

*Opening-Boolean-phrase* | *Closing-Boolean-phrase* → *Capitalized-phrase*

*Condition-sentence* → *Process-name* occurs if *Boolean-assertion*

*Negative-condition-sentence* → *Process-name* occurs if *Negated-Boolean-assertion*

*Compound-condition-sentence* → *Positive-process-name* occurs if *Boolean-assertion* /comma otherwise *Negative-process-name* occurs

*Case-sentence* → *(Case-phrase/comma)<sup>2+</sup>* and *Case- phrase*

| *Case-phrase* and *Case- phrase*

*Case-phrase* → In case *Object-name* is *State-name/comma* *Process-name* occurs

*Positive-process-name* | *Negative-process-name* → *Process-name*

*Complexity-management-sentence* → *Diagram-expansion-sentence*

| *Detailing-sentence* | *Abstracting-sentence*

*Diagram-expansion-sentence* → *Single-thing-expansion-sentence*

| *Many-things-expansion-sentence*

*Single-thing-expansion-sentence* → *Thing-name* in *Diagram-id* is expanded in *Diagram-id*

(|| /comma while (*Thing-name* is condensed | *Thing-list* are condensed))?

*Many-things-expansion-sentence* → *Thing-list* in *Diagram-id* are expanded in *Diagram-id*

(|| /comma while (*Thing-name* is condensed | *Thing-list* are condensed))?

*Thing-name* → *Object-name* | *Process-name*

*Thing-list* → *Object-list* | *Process-list*

| *Object-list* along with *Process-list* | *Process-list* along with *Object-list*

*Diagram-id* → SD || (/dash integer)\*

*Detailing-sentence* → *Process-detailing-sentence*

| *Object-detailing-sentence*

| *State-detailing-sentence*

*Process-detailing-sentence* → *Process-name* zooms into *Thing-list*